CodeZero
代码分析

徐伟健

src/lib目录

- 概况：内核用到的基本库函数
  - 位操作（bit.c）
  - 标识符池（idpool.c）
  - 内存缓存（memcache.c）
  - 互斥量（mutex.c）
  - 格式化输出（printk.c）
  - 单字符输出（putc.c）
  - 字符串处理（string.c）
  - 线程等待与唤醒（wait.c）
- 特点：尽量保持简单

位操作（bit.c）

- __clz　　数给定数前面有多少连续二进制0
- find_and_set_first_free_bit
  找给定内存区域第一个不为0的位
- check_and_clear_bit
  检测给定内存区域某位是否为1并清除
- check_and_set_bit
  检测给定内存区域某位是否为0并置1
- setbit　　设置某些位为1
- clrbit　　清除某些位（清除为置0）
- tstbit　　检测某些位是否为1
- tstclr　　检测某些位是否为1，并清除那些位

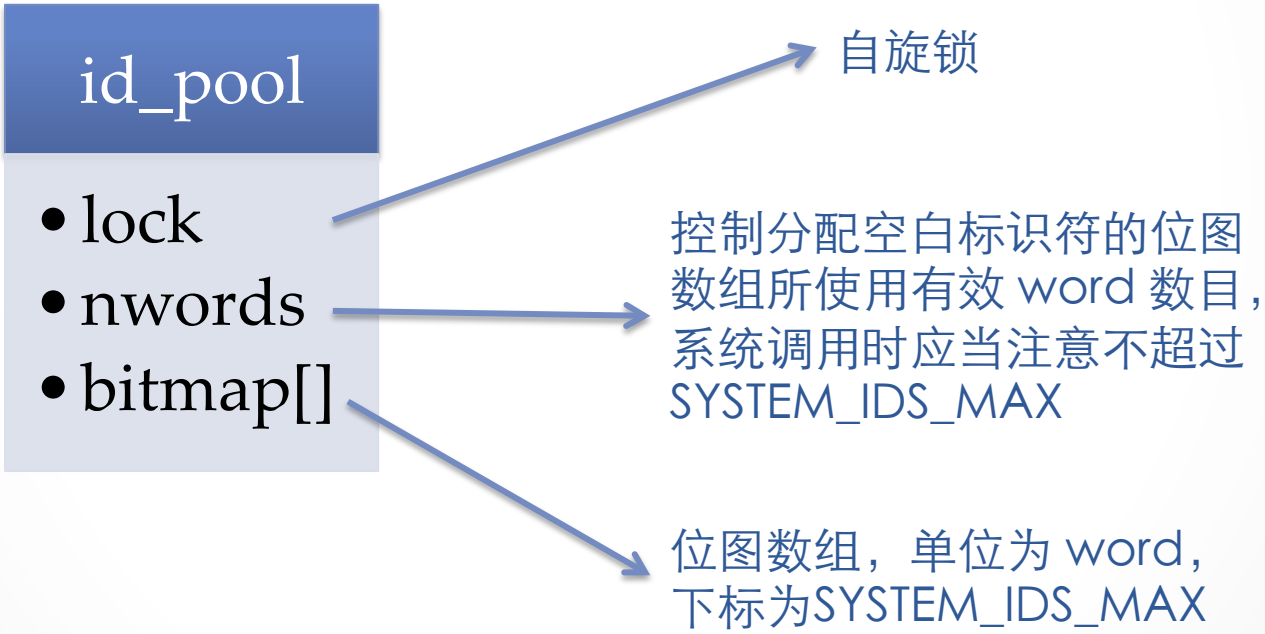
注：这里加黑的三个函数主要用于位图（bitmap）的操作

标识符池（idpool. c）

- id_pool_new_init　　初始化标识符池
- id_new　　获取池中一空白标识符
- id_del　　删除池中给定标识符
- id_get　　获取指定空白标识符

注：这里的标识符（id）主要是用于线程（thread id）以及用户空间的标识

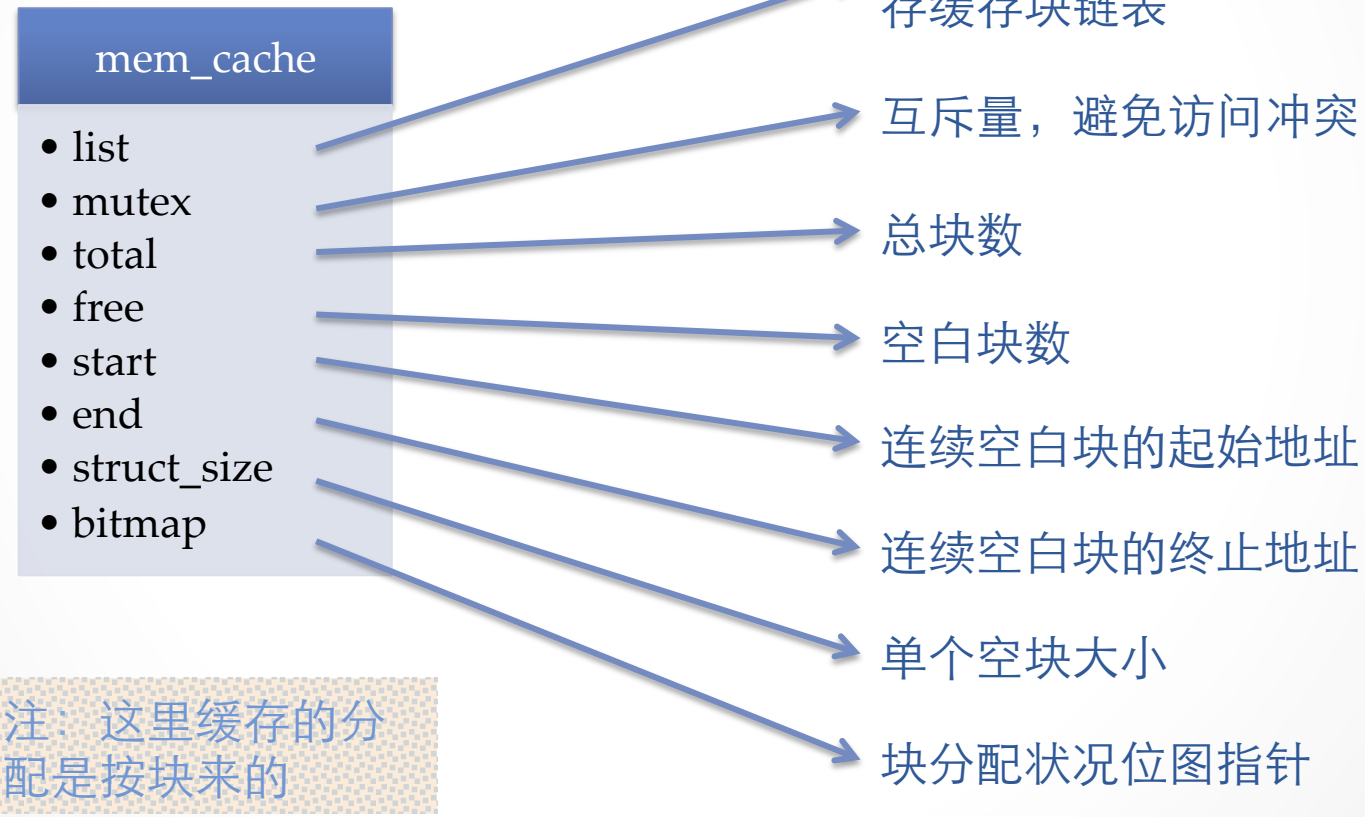
标识符池（idpool. c）



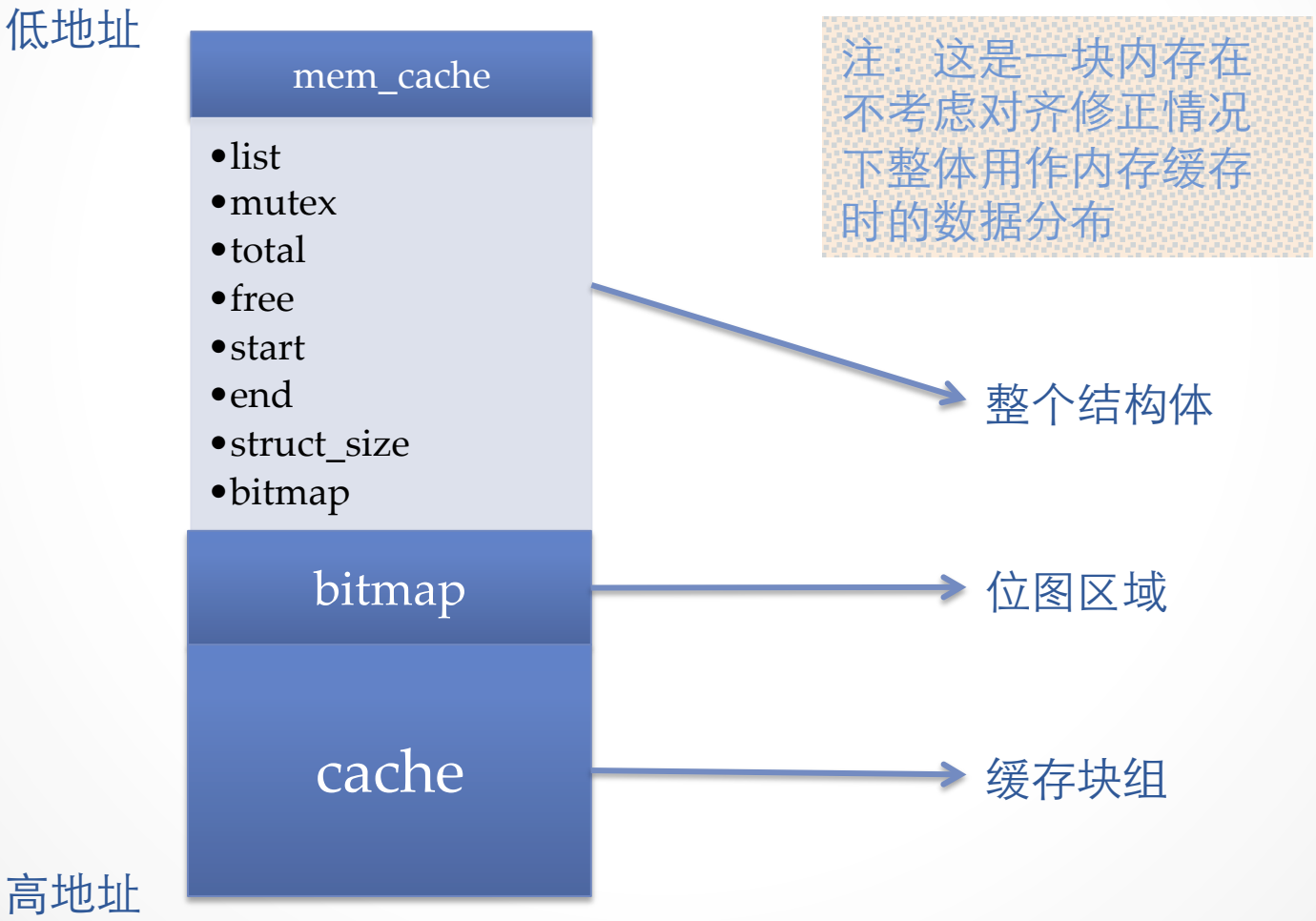
内存缓存（memcache.c）

- mem_cache_bufsize
  计算 memcache 结构总共需要空间大小
- mem_cache_zalloc
  从 cache 里寻找并分配一块空白块，并初始化为0
- mem_cache_alloc
  从 cache 里寻找并分配一块空白块
- mem_cache_free
  清除给定 memcache 里某一占用的块
- mem_cache_is_full　　判断 memcache 是否满
- mem_cache_is_empty　　判断 memcache 是否空
- mem_cache_is_last_free　　判断 memcache 是否只剩 1 个
- mem_cache_total_empty　　返回 memcache 中空白块的数目

内存缓存（memcache.c）



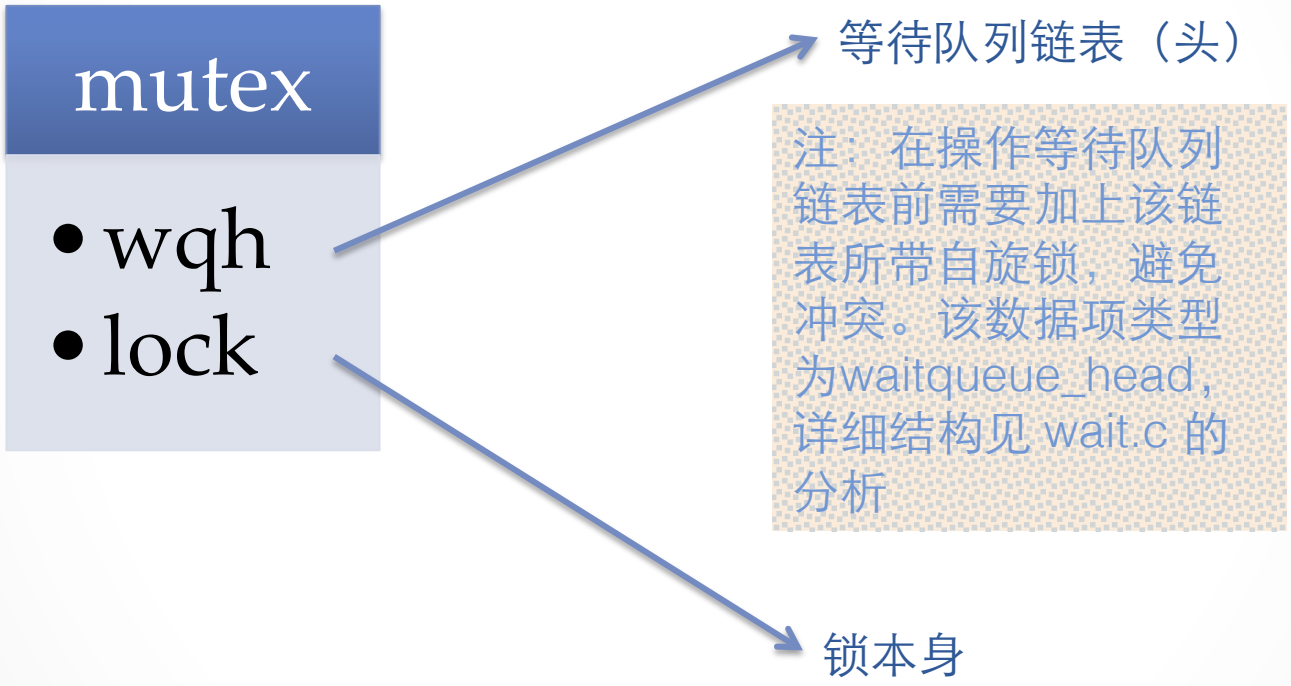
内存缓存（memcache.c）



互斥量（mutex.c）

- mutex_trylock　　非阻塞加锁
- mutex_lock　　阻塞加锁
- mutex_unlock　　同步解锁
- mutex_unlock_async　　异步解锁

互斥量（mutex.c）


mutex
• wqh
• lock
等待队列链表（头）
注：在操作等待队列链表前需要加上该链表所带自旋锁，避免冲突。该数据项类型为waitqueue_head，详细结构见 wait.c 的分析
锁本身

互斥量（mutex. c）

```
/* Non-blocking attempt to lock mutex */
// 非阻塞方式加锁
// 锁住 mutex 成功返回 1，失败返回 0
int mutex_trylock(struct mutex *mutex)
{
        int success;
        //加自旋锁锁住等待队列
        spin_lock(&mutex->wqh.slock);
        ////////////////////////////////////
        //检测并加锁(test & lock)
        if ((success = __mutex_lock(&mutex->lock)))
                //锁的数目增加
                current->nlocks++;
        ////////////////////////////////////
        //解除自旋锁
        spin_unlock(&mutex->wqh.slock);

        return success;
}
```

格式化输出（printk.c）

格式化输出相关函数，具体功能如下：

- printk　　输出指定格式字符串

“%”表示特殊选项开始（而双写符号即 %% 则表示将其转义成普通“%”输出）

其格式满足

%[prefix][type]

其中prefix表示前缀的附加选项，type表示数据类型

格式化输出（printk. c）

其中 type 有以下几种（注意，其中d和u的使用时，prefix中只有宽度参数有效）

- c　　单个字符
- m　　64位变量的十六进制值
- d　　整数
- u　　无符号整数
- x　　十六进制数
- s　　字符串
- p　　精度为word十六进制长度输出十六进制数

（即精度为16位十六进制数，以下统称为标准长度，常用于内存地址、32位数据等的显示）

注：精度在这里十六进制数的输出中定义为数据的最小输出长度，若不足则添加前导‘0’，相当于右对齐加前导0加指定宽度的效果。

格式化输出（printk. c ）

而 prefix 有以下几种，可以看做是

[-][width][w][.[precision][w]]

- ‘-’ 表示右对齐
- width 是一个十进制数字，表示输出宽度，有前导0（即数字以0开头）时设置使用‘0’填充空余占位字符，没有前导0时使用空格‘’填充空余占位字符
- w(第一个) 表示设定宽度为标准长度
- ‘.’ 表示之后还设定精度
- precision 是一个十进制数字，表示输出精度
- w(第二个) 表示设定精度为标准长度

单字符输出（putc. c）

功能简单，输出单字符

- putc　　输出单个字符

- 代码中包含了换行符转义，适用于不同硬件平台

字符串处理（string. c）

- memset　写入一块内存区域所有字节为某值
- memcpy　拷贝一块内存区域到指定位置
- strcmp　比较两个字符串大小
- strncpy　拷贝字符串前部分字节

注：memset 和 memcpy 函数为了优化速度使用汇编实现，与体系结构有关

线程等待与唤醒（wait.c）

- task_set_wqh　　设置 KTCB 中等待队列参数
- task_unset_wqh　　清除 KTCB 中等待队列参数
- wait_on_prepared_wait　　线程已做好等待准备时的调度
- wait_on_prepare　　线程等待准备
  （需结合 wait_on_prepared_wait 一起使用）
- wait_on　　直接线程等待
- wake_up_all　　唤醒所有线程
- wake_up　　唤醒一个线程
- wake_up_task　　唤醒一个给定线程

注：wait_on_prepare 和 wait_on_prepared_wait 一起使用是线程等待的一种方法，wait_on 则是另一种。区别在于前者有关抢占开抢占的过程而后者无